

PENYELESAIAN MASALAH SYARAT BATAS PERSAMAAN DIFERENSIAL BIASA DALAM SCILAB DENGAN MENGUNAKAN BVODE

**A. D. GARNADI¹, F. AYATULLAH¹, D. EKAstrya¹, S. NURDIATI¹,
M.T. JULIANTO¹, W. ERLIANA¹**

Abstrak

Diuraikan penggunaan rutin bvode di lingkungan SCILAB untuk menyelesaikan masalah syarat batas sistem persamaan diferensial biasa. Tujuannya ialah agar pengguna dapat mempergunakan bvode setelah membaca uraian penggunaannya. Penggunaan rutin digambarkan dengan tiga contoh yang memperlihatkan kegunaannya.

PENDAHULUAN

Persamaan Diferensial Biasa (PDB) sering muncul sebagai model permasalahan dalam berbagai bidang ilmu pengetahuan. Pencarian solusi PDB diperlukan untuk memperoleh interpretasi dari model permasalahan semula. Solusi analitik dari suatu PDB tidak selalu mudah diperoleh. Pada umumnya, untuk mencari salah satu solusi yang dikehendaki dari suatu sistem PDB, diperlukan suatu nilai awal yang memenuhi sistem PDB tersebut pada suatu titik $x = a$. Permasalahan seperti ini dikenal dengan nama Masalah Nilai Awal (MNA), yang secara ringkas dapat dinyatakan oleh:

$$\begin{aligned} y^n &= f(x, y, y', y'', \dots, y^{n-1}) \\ y(a) &= b \end{aligned} \tag{1}$$

Ada kalanya solusi yang sangat spesifik ingin didapatkan dari suatu sistem PDB dengan cara menentukan nilai-nilai awal pada lebih dari satu titik x . Permasalahan seperti ini dikenal dengan nama Masalah Syarat Batas (MSB), yang dapat dinyatakan oleh :

$$\begin{aligned} y^n &= f(x, y, y', y'', \dots, y^{n-1}) \\ y(a_1) &= b_1 \\ y(a_2) &= b_2 \end{aligned} \tag{2}$$

Rutin numerik yang mampu menyelesaikan MNA telah banyak dikembangkan, namun sedikit yang mampu menyelesaikan MSB. Sebuah rutin untuk menyelesaikan MSB berdasarkan metode kolokasi disediakan dalam lingkungan Scilab, rutin ini bernama bvode. Metode kolokasi juga digunakan dalam lingkungan MATLAB untuk rutin bvp4c yang disediakan.

¹Departemen Matematika, Fakultas Ilmu Pengetahuan Alam, Jalan Meranti Kampus IPB Dramaga Bogor, 16680.

Tulisan ini merupakan sebuah tutorial yang memberikan penuntun bagaimana memformulasikan masalah (menyusun perintah), mendapatkan solusi numerik, dan menggambarkan solusi secara grafis dari MSB dengan menggunakan rutin `bvode`. Tulisan ini juga merupakan studi pendahuluan numerik atas ketersediaan lingkungan pemecah masalah numerik (*numerical problem solving environment*{PSE}) yang bersifat *open-source* sebagai alternatif dari PSE komersial populer MATLAB yang menawarkan rutin `bvp4c` untuk memecahkan MSB.

Salah satu tujuan tulisan ini bersifat pedagogis yang menguraikan secara sederhana bagaimana memperoleh solusi numerik MSB sebuah PDB. Dengan demikian, seorang pemula dapat mempergunakan `bvode` untuk menyelesaikan MSB yang dihadapinya tanpa kesulitan. Selain itu, contoh yang diberikan muncul dari masalah perkuliahan yang biasanya ditemui di tahun ketiga atau keempat di perguruan tinggi. Selain itu, tulisan ini bertujuan untuk menyediakan dokumentasi tertulis berbahasa Indonesia. Tujuan lainnya ialah memberikan teladan penggunaan *Open Source* di lingkungan komputasi matematika di Indonesia, yang secara tidak langsung memberi dukungan pada proyek nasional IGOS (*Indonesia Goes Open Source*).

Tulisan ini disusun dengan urutan sebagai berikut. Pertama diuraikan deskripsi dari `bvode`. Kemudian diperlihatkan penggunaannya untuk tiga buah contoh diluar contoh yang diberikan dalam lingkungan Scilab tentang bagaimana mempergunakan `bvode` untuk menyelesaikan sebuah MSB. Kemudian diakhiri oleh kesimpulan serta kemungkinan pekerjaan lanjutan.

DESKRIPSI `bvode`

Rutin `bvode` yang tersedia dalam PSE Scilab, merupakan perangkat simulasi untuk menyelesaikan MSB. Rutin `bvode` pada dasarnya mempergunakan pustaka COLNEW yang merupakan perbaikan dari rutin program COLSYS ([1], [2]) yang ditulis mempergunakan FORTRAN dan berlandaskan metode kolokasi. Tidak seperti *solver* PDB pada umumnya, `bvode` tidak mengharuskan pengguna mengubah persamaan diferensial orde tinggi menjadi sistem persamaan diferensial orde satu. Selain itu, dimungkinkan diberikannya sejumlah syarat di sejumlah titik ζ_j .

Bentuk MSB yang diasumsikan oleh `bvode` ialah:

$$\frac{d^{m_i} y_i}{dx^{m_i}} = f_i \left(x, y(x), \frac{dy}{dx}, \dots, \frac{d^{m_i-1} y_i}{dx^{m_i-1}} \right), \quad 1 \leq i \leq n_c, \quad (3)$$

$$g_j \left(\zeta_j, y(\zeta_j), \dots, \frac{d^{m_*} y}{dx^{m_*}}(\zeta_j) \right) = 0, \quad j = 1, \dots, m_*. \quad (4)$$

Dengan ζ_j merupakan posisi di mana syarat batas berlaju dan $a_L \leq \zeta_j \leq a_R$. Agar notasi tidak menyulitkan, tuliskan

$$m_* = m_1 + m_2 + \dots + m_{n_c},$$

$$z(y) = \left[y, \frac{dy}{dx}, \dots, \frac{d^{m_*}y}{dx^{m_*}} \right].$$

Dengan demikian, bentuk umum MSB yang diasumsikan oleh bvode ialah

$$\frac{d^{m_i}y_i}{dx^{m_i}} = f_i(x, z(y(x))), \quad 1 \leq i \leq n_c, \quad a_L \leq \zeta_j \leq a_R, \quad (5)$$

$$g_j(\zeta_j, z(y(\zeta_j))) = 0, \quad j = 1, \dots, m_*. \quad (6)$$

Rutin bvode memiliki kemampuan untuk menyelesaikan MSB yang linear maupun non-linear. Karena itu rutin ini mengharuskan pengguna menyusun sendiri matriks Jacobian dari PDB yang hendak diselesaikan, maka untuk beberapa masalah, tingkat kerumitan yang paling besar akan terasa pada penyusunan matriks Jacobian ini.

Seperti halnya fungsi lain pada SCILAB, rutin bvode memiliki tata cara pemanggilan sebagai berikut ini

```
[ z ] = bvode(points, ncomp, m, aleft, aright, zeta, ipar, ...
ltol, tol, fixpnt, fsub, dfsub, gsub, dgsub, guess);
```

dengan z merupakan vektor baris yang berisi solusi numerik dari MSB yang ingin diselesaikan. Komponen $z(i, :)$ merepresentasikan turunan ke- $(i-1)$ dari solusi pada selang domain. Sebagai contoh, $z(1, :)$ menyatakan y , $z(2, :)$ menyatakan y' , dan seterusnya hingga $z(m^*, :)$ menyatakan $y^{(m^*-1)}$.

Lima argumen terakhir yang diperlukan bvode berbentuk fungsi yang harus didefinisikan sendiri oleh pengguna, sedangkan argumen lainnya merupakan informasi yang diperlukan bvode untuk dapat mencari solusi numerik berkaitan dengan MSB yang ingin diselesaikan. Berikut adalah penjelasan kegunaan masing-masing parameter yang digunakan oleh bvode.

z	: Solusi dari PDB yang dievaluasi atas mesh yang diberikan oleh <code>points</code> .
<code>Points</code>	: Array yang menyimpan titik di selang domain dimana MSB akan dicari solusinya.
<code>Ncomp</code>	: Jumlah persamaan diferensial. Syaratnya haruslah <code>ncomp</code> ≤ 20.
<code>M</code>	: Vektor dengan panjang <code>ncomp</code> , yang isinya <code>m(j)</code> berupa orde dari persamaan diferensial ke- j . Orde dari persamaan diferensial harus disyaratkan <code>m(j)</code> ≤ 4.
<code>Aleft</code>	: Ujung kiri dari selang domain dimana y didefinisikan.
<code>Arigh</code>	: Ujung kanan dari selang domain dimana y didefinisikan.
<code>zeta</code>	: Isi <code>zeta(j)</code> berisi titik syarat (batas) tambahan ke- j . Isi harus diurutkan sehingga <code>zeta(j)</code> ≤ <code>zeta(j+1)</code> . Semua titik syarat tambahan (batas) harus pada titik mesh di semua mesh yang digunakan. Khususnya, titik

tersebut harus merupakan bagian dari mesh awal. Perhatikan pula uraian lebih rinci mengenai parameter `ipar(11)` dan `fixpnt` di bawah ini.

`ipar` : Array berentri bilangan bulat dengan ukuran sedikitnya 11. Daftar parameter dari `ipar` akan diuraikan kemudian.

`ltol` : Array dengan panjang yang ditentukan `ipar(4)`. `ltol(j)`=1 menentukan toleransi ke- j di tol mengendalikan error di komponen ke- l dari $z(u)$. Juga dipersyaratkan bahwa $1 \leq \text{ltol}(1) < \text{ltol}(2) < \dots < \text{ltol}(\text{ntol}) \leq \text{mstar}$.

`tol` : Array dengan panjang yang ditentukan `ipar(4)`. `tol(j)` menyatakan toleransi pada komponen ke `ltol(j)` dari solusi $z(u)$.

`fixpnt` : Titik selain `aleft` dan `aright` yang akan dimasukkan ke dalam mesh.

`fsub` : Fungsi yang mendefinisikan PD yang ingin dicari solusi numeriknya.

`dfsub` : Matriks Jacobian dari `fsub`.

`gsub` : Fungsi yang mendefinisikan persamaan syarat batas untuk MSB yang bersangkutan.

`dgsub` : Matriks Jacobian dari `gsub`.

`guess` : Tebakan awal bagi solusi (jika tersedia).

Dari kelima belas parameter di atas, argumen `ipar` adalah argumen yang paling panjang. Argumen ini memiliki 11 komponen yang secara keseluruhan memberi kesempatan pengguna untuk mengatur setting pada metode numerik yang digunakan `bvode`. Sebagian komponen dari parameter `ipar` berhubungan langsung dengan beberapa argumen `bvode` pada daftar parameter di atas. Sebagian besar dari komponen-komponen tersebut memiliki nilai default nol. Karena jumlah komponennya yang terbilang cukup banyak, ada baiknya jika dibuat variabel berdimensi 1×11 yang berisi nilai nol (default) sebelum mengisi komponen-komponen yang tidak bernilai default, yaitu

```
ipar = zeros(1,11);.
```

Isikan masing-masing komponen parameter `ipar` berdasarkan keterangan yang didaftarkan di bawah ini:

`ipar(1)` : = 0 jika MSB linear dan = 1 jika MSB non-linear.

`ipar(2)` : Banyaknya titik kolokasi per subinterval ($=k$) dimana orde maksimum $PD \leq k \leq 7$. Jika `ipar(2)=0` maka secara default $k = \max(\max m(i)+1, 5-\max m(i))$.

`ipar(3)` : Banyaknya subinterval pada mesh awal ($=n$). Jika `ipar(3) = 0` maka secara default $n = 5$.

`ipar(4)` : Banyaknya komponen solusi beserta turunannya yang diberi toleransi ($=\text{ntol}$), dengan aturan $0 \leq \text{ntol} \leq m^*$.

`ipar(5)` : Nilai yang menentukan banyaknya subinterval maksimum (n_{\max}) pada selang domain. Pilihlah `ipar(5)` berdasarkan rumus berikut: `ipar(5) \geq ipar(5).nsizéf` di mana `nsizéf = 4+3m^*+(5+kd).kdm+(2m^*-nrec).2m^*` dengan `nrec` adalah banyaknya syarat batas pada ujung kanan selang domain `ipar(5)` diperuntukkan evaluasi pada titik-titik selang real.

`ipar(6)` : Nilai yang menentukan banyaknya subinterval maksimum n_{\max} pada

selang domain. Pilihlah `ipar(6)` berdasarkan rumus berikut:
`ipar(6) ≥ nmax.nsizei` di mana `nsizei = 3+kdm` dengan
`kdm=kdm+m*`; `kd = k.ncomp`; `ipar(6)` diperuntukkan evaluasi pada
titik-titik integer.

`ipar(7)` : Kontrol output berikan nilai berikut: = -1 untuk printout diagnostik
penuh = 0 untuk printout sederhana. = 1 tanpa printout.

`ipar(8)` : = 0 untuk mesh awal seragam.

`ipar(9)` : = 0 jika tidak tersedia tebakan awal bagi solusi. = 1 jika tebakan awal
tersedia pada fungsi `guess`.

`ipar(10)` = 0 jika masalah bersifat reguler. = 1 jika iterasi nonlinear tidak
bergantung pada kekonvergenan dari iterasi sebelumnya (hanya
digunakan jika `ipar(1) = 1`). = 2 jika proses ingin dihentikan setelah
(a)terjadi dua kali nonkonvergen berturut-turut, atau (b)pendekatan
galat didapatkan untuk pertama kalinya.

`ipar(11)` : Banyaknya titik selain ujung-ujung selang domain yang akan
dimasukkan ke dalam mesh (dimensi dari argumen `fixpnt`).

CONTOH

Pada bagian ini akan dibahas tiga contoh bagaimana menyelesaikan MSB dengan mempergunakan `bvode`. Pada bagian ini juga akan ditunjukkan bagaimana cara menggambar masing-masing komponen solusi numerik yang didapat dari `bvode`.

Contoh 1. Dalam contoh ini diberikan ilustrasi bagaimana mencari solusi numerik dari MSB yang melibatkan parameter yang tidak diketahui. Permasalahannya ialah menghitung nilai eigen dari persamaan Mathieu berikut ini,

$$y'' + (\lambda - 2q \cos(2x))y = 0 \quad (7)$$

pada selang $[0, \pi]$, dengan syarat batas $y'(0) = 0$, $y'(\pi) = 0$ untuk $q = 5$. Solusi dinormalisasi dengan cara menetapkan solusi memenuhi $y(0) = 1$. Permasalahan sesungguhnya ialah mencari nilai λ yang memenuhi syarat batas $y'(\pi) = 0$. Nilai tebakan awal bagi λ menjadi keharusan dalam menyelesaikan masalah ini. MSB di atas diimplementasikan dalam SCILAB sebagai berikut ini. Persamaan diferensialnya didefinisikan dengan skrip berikut.

```
function f=fsub(x, zu)
f=-(lambda-2*5*cos(2*x))*zu(1);
endfunction
```

Untuk persamaan di atas, diperlukan matriks Jacobian yang ditulis dengan skrip seperti berikut ini.

```
function df=dfsub(x, zu)
df=[-(lambda-2*5*cos(2*x)), 0];
endfunction
```

Sementara syarat batasnya diberikan oleh:

```
function g=gsub(i, zu),
    select i
    case 1 then
g=zu(1)-1
    case 2 then
g=zu(2)
    case 3 then
g=zu(2)
    end
endfunction.
```

Untuk syarat batas tersebut di atas, diperlukan matriks Jacobian, yaitu

```
function dg=dgsub(i, z)
    select i
    case 1 then
dg=[1,0]
    case 2 then
dg=[0,1]
    case 3 then
dg=[0,1]
    end
endfunction.
```

Berikut sejumlah parameter yang diperlukan, yaitu

```
fixpnt=[ ];
ncomp=1; nrec=1;
k=4;
nmax=200;
kd=N*ncomp;
kdm=kd+M;
nsizei=kdm+3;
ip6=nmax*nsizei;
nsizef=4+3*M+(5+kd)*kdm+(2*M-nrec)*2*M;
ip5=nmax*nsizef;
ipar=[001 2ip5ip6100 0 0]
ltol=1:2;
tol=[1.e-5,1.e-5];
```

Karena tebakan awal bagi solusi y tidak tersedia maka fungsi guess didefinisikan seperti di bawah ini.

```
function [zu, mpar]=guess(x)
zu=0;
mpar=0;
endfunction
```

Jika tebakan awal bagi nilai eigen dipilih $\lambda = 15$, maka untuk menyelesaikan MSB pada persamaan (7) sekaligus mendapatkan nilai pendekatan untuk λ dapat dilakukan dengan menambahkan *loop* seperti pada *listing* berikut,

```
lambda_0=15;
xpoints=x_low:0.001:x_up;
```

```

for j=0:0.001:20000
    lambda=lambda_0+j;

    zu=bvode(xpoints,N,m,x_low,x_up,zeta,ipar,ltol,tol,fixpnt,fs
    ub,dfsub,gsub,dgsub,guess);
    if abs(zu(2,$))<j
        break;
    end;
end

```

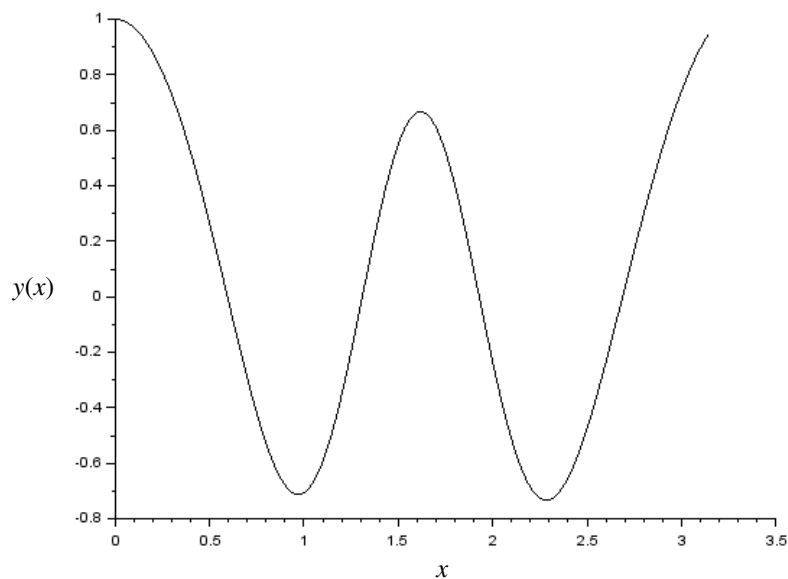
Untuk melihat gambar grafik solusi numerik untuk $y(x)$ yang dihasilkan bvode dan untuk mengetahui nilai λ yang didapat dari *loop* di atas, perintah berikut dengan cepat akan menampilkan keduanya:

```

lambda
plot2d(xpoints,zu(1,:));

```

Dari hasil perhitungan yang dilakukan oleh bvode beserta *loop* di atas, didapat nilai pendekatan $\lambda = 16.058$ dan gambar grafik solusi $y(x)$ seperti yang terlihat pada Gambar 1. Perhitungan di atas mempergunakan mesh dengan lebar kisi $\Delta x = 0.001$ yang seragam untuk selang $[0, \pi]$.



Gambar 1 Fungsi eigen dari persamaan Mathieu terkait dengan nilai eigen yang digunakan dalam artikel

Contoh 2. Pada contoh berikut diberikan sebuah MSB yang berasal dari Kontrol Optimal. Biasanya, syarat cukup untuk kontrol optimal berupa MSB, yang biasanya lebih mudah menyelesaikannya untuk banyak masalah sederhana.

Perhatikan masalah sistem terkontrol

$$y' = y^2 + v$$

dan kita inginkan kontrol v mengendalikan lintasan dari 2 pada saat $t = 0$ ke -1 pada saat $t = 10$. Dengan demikian, kita inginkan v meminimumkan ongkos kuadrat berikut

$$J(y, v) = \int_0^{10} (y^2 + 10v^2) dt.$$

Fungsi ongkos ini menyebabkan variabel keadaan y dan variabel kontrol v terpenalisasi, sehingga menyebabkan variabel keadaan dan kontrol menjadi kecil.

Syarat cukup dapat diperoleh dengan mendefinisikan fungsi Hamiltonian

$$H = (y^2 + 10v^2) + \lambda(y^2 + v)$$

dan kita tuliskan masalah syarat batas persamaan diferensial aljabar

$$\begin{aligned} y' &= \frac{\partial H}{\partial \lambda}, \\ \lambda' &= \frac{\partial H}{\partial y}, \\ 0 &= \frac{\partial H}{\partial v}, \end{aligned} \tag{8}$$

$$y(0) = 2, y(10) = 1,$$

Bila v dieliminasi, akan diperoleh MSB

$$\begin{aligned} y' &= y^2 + v, \\ \lambda' &= -2y - 2\lambda y, \\ 0 &= 20v + \lambda, \\ y(0) &= 2, y(10) = 1, \end{aligned} \tag{9}$$

MSB di atas diimplementasikan dalam SCILAB sebagai berikut ini. Persamaan diferensialnya didefinisikan dengan skrip berikut.

```
function w = f(x, z)
    w1 = z(1)^2 - z(2)/20 ;
    w2 = 2*z(1) - 2*z(1)*z(2) ;
    w = [w1 ; w2] ;
endfunction
```

Untuk persamaan di atas, diperlukan matriks Jacobian yang ditulis dengan skrip seperti berikut ini.

```
function w = df(x, z)
    w = [2*z(1), -1/20; -2-2*z(2), -2*z(1)];
endfunction
```

Sementara syarat batasnya diberikan oleh:

```
function w = g(i, z)
    if i==1 then w=z(1)-2,
    else w=z(1)+1;
end
```



```
endfunction.
```

Untuk syarat batas tersebut di atas, diperlukan matriks Jacobian, yaitu

```
function w=dg(i,z)
    w=[1 0];
endfunction.
```

MSB di atas, diselesaikan di atas selang $[0,10]$ dengan lebar kisi $\Delta t = 0.1$ yang seragam, berikut sejumlah parameter yang diperlukan, yaitu

```
tt=0:0.1:10 ;
ncomp=2;nrec=1;
m=[1 1]; k=4;
nmax=200;
mstar=2;
kd=k*ncomp;
kdm=kd+mstar;
nsizei=kdm+3;
ip6=nmax*nsizei;
nsizef=4+3*mstar+(5+kd)*kdm+(2*mstar-nrec)*2*mstar;
ip5=nmax*nsizef;
zeta = [0 10];
ipar = [1 0 0 2, ip5, ip6, 0 0 0 0 0];
er=1.0d-3;
ltol=1:2;
tol = [er,er];
```

dan kita berikan fungsi tebakan awal, yaitu

```
function w =gs(x)
    w = [0,0;0,0];
endfunction.
```

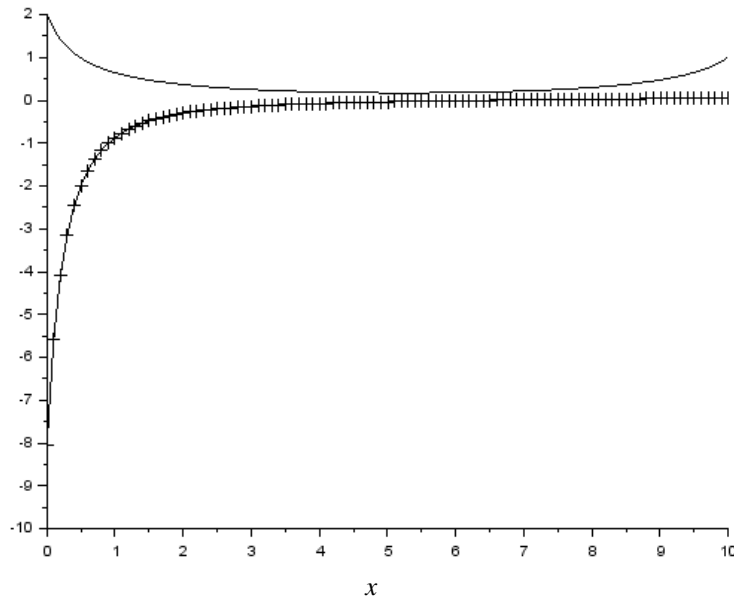
Kemudian untuk memperoleh solusi numerik MSB, diberikan dalam skrip SCILAB berikut

```
zz = bvode(tt,2,m,0,10,zeta,ipar,ltol,tol,0,f,df,g,dg,gs);
control=-(1/20)*zz(2,:);
```

Solusi numerik yang diperoleh, ditampilkan oleh Scilab dengan memberikan skrip berikut ini

```
plot2d(tt,zz(1,:));
plot2d(tt,control);
plot2d(tt,control,style=-1);
```

yang tampilan grafisnya diperlihatkan pada Gambar 2.



Gambar 2 Optimal y (garis) dan kontrol v (garis- x) untuk contoh 2.

Contoh 3. Pada contoh berikut diberikan sebuah MSB yang berasal dari formula kredibilitas yang memungkinkan aktuaris untuk menyeimbangkan dua hal, yaitu keakuratan dan kelinearan [9]. Berikut merupakan MSB untuk kasus di mana sebaran marginal (f) dari besarnya klaim (X) ialah tak nol pada suatu interval terbatas $[a, b]$, lihat Young [10].

$$\frac{h}{f(x)} [f(x)d''(x)]'' + d(x) = \mu(x), \quad (10)$$

dengan

$$d''(a) = d''(b) = d'''(a) = d'''(b) = 0.$$

Pada contoh ini, misalkan $f(x) = 1$ pada interval $[0, 1]$, $h = 0.0025$, dan

$$\mu(x) = \frac{2}{(x+2)\ln 1.5} - 2 + \frac{2}{(x+2)} \int_1^{0.5x+1} \frac{3-2\theta}{\ln 1.5 - \ln \theta} d\theta.$$

Misalkan pula

$$v(x) = \int_1^{0.5x+1} \frac{3-2\theta}{\ln 1.5 - \ln \theta} d\theta,$$

sehingga

$$v'(x) = \frac{1-x}{2[\ln 1.5 - \ln(0.5x+1)]}.$$

Dengan demikian, persamaan (10) menjadi

$$0.0025d''''(x) + d(x) = \frac{2}{(x+2)\ln 1.5} - 2 + \frac{2}{(x+2)} v(x)$$

$$d''''(x) = \frac{1}{0.0025} \left[\frac{2}{(x+2) \ln 1.5} - 2 + \frac{2}{(x+2)} v(x) - d(x) \right].$$

Jadi, MSB untuk kasus tersebut ialah

$$v'(x) = \frac{1-x}{2[\ln 1.5 - \ln(0.5x+1)]},$$

$$d''''(x) = \frac{1}{0.0025} \left[\frac{2}{(x+2) \ln 1.5} - 2 + \frac{2}{(x+2)} v(x) - d(x) \right],$$

dengan

$$v(0) = d''(0) = d''(1) = d'''(0) = d'''(1) = 0.$$

MSB di atas diimplementasikan dalam SCILAB sebagai berikut. Persamaan diferensialnya didefinisikan dengan skrip berikut.

```
function f=fsub(x, zu)
f1=(1-x)/(log(1.5)-log(1/2*x+1));
f2=(1/0.0025)*(2/((x+2)*log(1.5))+2/(x+2)*zu(1)-2-zu(2));
f=[f1; f2];
endfunction
```

Untuk persamaan di atas, diperlukan matriks Jacobian yang ditulis dengan skrip seperti berikut ini.

```
function df=dfsub(x, zu)
df=[0,0,0,0,0; 1/((x+2)), -1/0.0025,0,0,0];
endfunction
```

Sementara syarat batasnya diberikan oleh:

```
function g=gsub(i, zu),
select i
case 1 then
g=zu(1)
case 2 then
g=zu(4)
case 3 then
g=zu(5)
case 4 then
g=zu(4)
case 5 then
g=zu(5)
end
endfunction
```

Untuk syarat batas tersebut, diperlukan matriks Jacobian, yaitu

```
function dg=dgsub(i, z)
select i
case 1 then
dg=[1,0,0,0,0]
case 2 then
dg=[0,0,0,1,0]
```

```
case 3 then
dg=[0,0,0,0,1]
case 4 then
dg=[0,0,0,1,0]
case 5 then
dg=[0,0,0,0,1]
end
endfunction
```

MSB di atas, diselesaikan di atas selang $[0,1]$ dengan lebar kisi $\Delta t = 0.01$ yang seragam, berikut sejumlah parameter yang diperlukan, yaitu

```
N=2;
m=[1 4];
M=sum(m);
x_low=0; x_up=1;

fixpnt=[ ];
xpoints=x_low:0.01:x_up;
ncomp=2; nrec=2;
m=[1 4]; k=4;
nmax=200;
kd=N*ncomp;
kdm=kd+M;
nsizei=kdm+3;
ip6=nmax*nsizei;
nsizef=4+3*M+(5+kd)*kdm+(2*M-nrec)*2*M;
ip5=nmax*nsizef;
ipar=[0,0,10,5,ip5,ip6,-1,0,0,0,0]
ltol=1:5;
tol=[1.e-11,1.e-11,1.e-11,1.e-11,1.e-11,1.e-11]
```

dan kita berikan fungsi tebakan awal, yaitu

```
function w=guess(x)
w=[0,0; 0,0];
endfunction
```

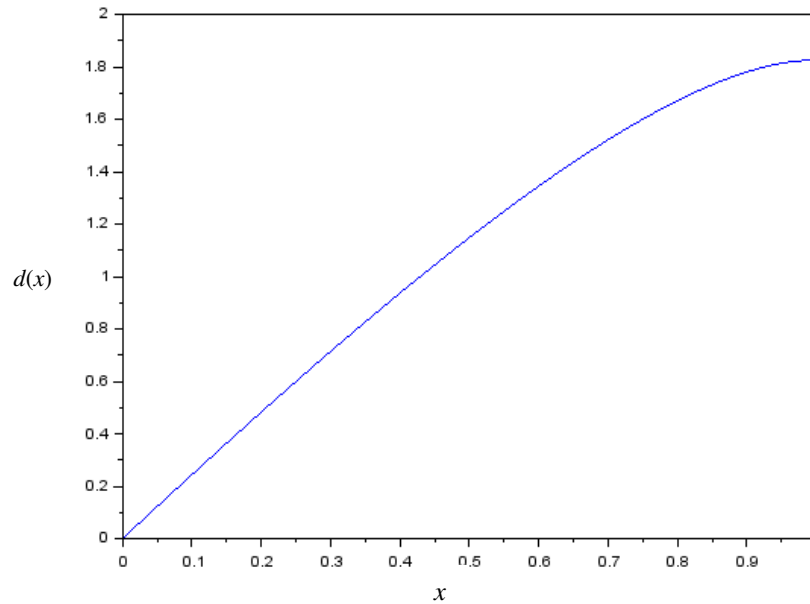
Kemudian untuk memperoleh solusi numerik MSB, diberikan dalam skrip SCILAB berikut

```
zu=bvode(xpoints,N,m,x_low,x_up,zeta,ipar,ltol,tol,fixpnt,fsu
b,dfsub,gsub,dgsub,guess)
```

Solusi numerik yang diperoleh, ditampilkan oleh Scilab dengan memberikan skrip berikut ini

```
plot(xpoints,zu(1,:)).
```

Tampilan grafisnya diperlihatkan pada Gambar 3.



Gambar 3 Solusi optimum (d) dengan $h = 0.0025$.

SIMPULAN

Masalah syarat batas sering muncul sebagai model dari berbagai bidang ilmu. Karena solusi analitik MSB tidak selalu tersedia, metode numerik menjadi salah satu cara untuk memperoleh solusinya. Selain itu, dengan tersedianya rutin yang memudahkan pengguna untuk menyelesaikan MSB secara numerik, dimungkinkan melakukan simulasi secara berulang untuk menguji berbagai skenario pemodelan. Tulisan ini mendemonstrasikan rutin simulasi MSB dalam lingkungan problem solving environment SCILAB yang bersifat *open source*. Perlu kiranya pengujian lebih lanjut atas rutin ini, seperti halnya dilakukan oleh [7] yang menguji rutin simulasi MSB di lingkungan komersial MATLAB. Pengujian beragam MSB dari berbagai bidang ilmu sebagaimana diberikan oleh [2], dapat diujikan di lingkungan SCILAB untuk bahasan lanjutan.

UCAPAN TERIMA KASIH

Penelitian ini dibiayai secara parsial oleh Hibah Penelitian PUPT-IPB melalui nomor kontrak : 083/SP2H/PL/Dit.Litabmas/II/2015. Terima kasih juga kepada DAAD atas bantuan literatur ilmiah berdasar kontrak no A/97/42703.

DAFTAR PUSTAKA

- [1] Ascher U, Christiansen J, Russel RD. 1981. Collocation Software for Boundary-Value ODEs. *ACM Transactions on Mathematical Software*. 7(2).
- [2] Ascher U, Mattheij RM, Russel RD, *Numerical Solution of Boundary Value Problem*. New Jersey: Prentice-Hall, Englewood Cliffs.
- [3] Campbell SL, Chancelier JP, Nikoukhah R. 2005. *Modeling and Simulation in Scilab/Scicos*. Springer.
- [4] Higham DJ, Higham NJ. 2000. *MATLAB Guide*. SIAM.
- [5] Kierzenka J, Shampine L. 2001. A BVP solver on residual control and the Matlab PSE. *ACM Transactions on Mathematical Software*. 27(3): 299-316.
- [6] Moler CB. 2004. *Numerical Computing with MATLAB*. SIAM.
- [7] Shampine L, Kierzenka J, Reichelt M. Solving Boundary Value Problems for Ordinary Differential Equations in Matlab with bvp4c, [http://www.mathworks.com/support/solutions/files/s8314/bvp_paper.pdf].
- [8] Layar bantuan SCILAB untuk rutin bode.
- [9] Kaas R, Goovaerts M, Dhaene J, Denuit M. 2009. Credibility theory. *Modern Actuarial Risk Theory: Using R*. 203-229.
- [10] Young VR. 1997. Credibility using a loss function from spline theory. *Scandinavian Actuarial Journal*. 1997(2): 160-185.